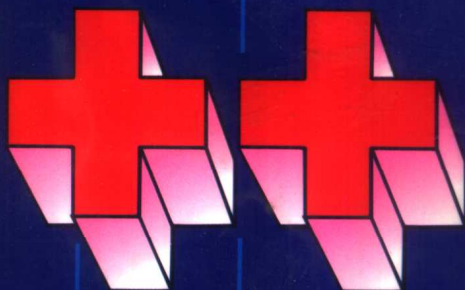


张基温 编著

0
++

程序设计基础 例题与习题



高等教育出版社

C++ 程序设计基础例题与习题

张基温 编著

高等教育出版社

(京)112 号

内 容 提 要

本书共收录了 387 道 C++ 的例题和习题。它们的选编基于三个方面:加深对基本语法现象的理解;系统地提高读者的程序设计能力;对语法知识作必要的扩充。这些例题和习题几乎覆盖了《C++ 程序设计基础》一书中出现的全部语法,并涉及了程序员所应了解和掌握的各种数据结构和算法。本书难易搭配、循序渐进,注重学生的能力培养和例题、习题的实用性。每个例题后都有几道稍作变化的习题,以供读者举一反三地练习使用。

本书虽作为《C++ 程序设计基础》一书的姊妹篇,但相对独立。可以作为 C++ 程序设计的习题课教材,也可供大学生和软件开发人员自学 C++ 时参考、自检。

图书在版编目(CIP)数据

C++ 程序设计基础例题与习题/张基温编者. - 北京:
高等教育出版社, 1997
ISBN 7-04-005948-7

I. C… II. 张… III. C 语言-程序设计-习题 IV.
TP312C-44

中国版本图书馆 CIP 数据核字(97) 第 05886 号

高等教育出版社出版

北京沙滩后街 55 号

邮政编码:100009 传真:64014048 电话:64054588

新华书店总店北京发行所发行

北京联华印刷厂印装

开本 787×1092 1/16 印张 17.5 字数 430 000

1997 年 5 月第 1 版 1997 年 5 月第 1 次印刷

印数 0 001—2 634

定价 18.00 元

凡购买高等教育出版社的图书,如有缺页、倒页、脱页等
质量问题者,请与当地图书销售部门联系调换

版权所有,不得翻印

前 言

面向对象的技术虽然已经提出十几年了,但直至目前才被广泛地接受。作为一种能承前启后的程序设计语言,C++ 正在走红。为满足这一历史性需求,高等教育出版社已于 1996 年 5 月出版了拙著《C++ 程序设计基础》。而这本以例题和习题形式出现的教材,是在编写《C++ 程序设计基础》的同时产生的一本孪生篇。它以例题为引导,每个例题之后给出几个类似的习题,以培养读者举一反三的能力。

《C++ 程序设计基础》的编写,虽然以尽快提高读者的程序设计能力为宗旨,但对初学者来说,在没有系统地掌握基本语法知识之前,空讲程序设计是很难接受的。因而它只能按语法体系进行组织,例题也着重于基本语法规则的演示或在已学语法基础上的编程训练。这样做,自然难以进行较系统的程序设计训练。作为《C++ 程序设计基础》的补充与扩充,本书在“章”一层上与《C++ 程序设计基础》一书相同,但“节”一层的组织上主要有以下三种类型:

(1) 以典型的算法或数据结构为标题。这种节中的例题和习题主要以系统地提高读者的解题能力为目的。

(2) 以某些重要语法为标题。这种节中的例题和习题一部分用于加深对重要语法的理解;另一部分从充分体现 C++ 的功能和风格的角度,作必要的语法知识扩充。

(3) 以“杂题”为标题。这种节中的例题和习题主要用于所在章中的零散语法现象的练习。

本书适合于作 C++ 的习题课教材,也可以作课外辅导教材自学使用。

教学试用证明,在学习《C++ 程序设计基础》的同时,按本书的内容进行认真的配合训练,C++ 进行程序设计的能力会有显著提高。

著名计算机教育家谭浩强教授对本书的编写十分关心,提出过重要的指导性意见。刘洪兆同学仔细地调试了书中的程序。本书在编写过程中还参考了有关著作。在本书出版之际,谨向各位帮助者致以衷心的感谢。

张基温

1996 年 6 月

责任编辑	刘建元
封面设计	王 喆
责任绘图	潘曙光
版式设计	马静如
责任校对	马静如
责任印制	常永年

参 考 文 献

- 1 Sharam Hermatpour 著. C程序的 C++ 指南. 阎龙译. 北京:北京航空航天大学出版社,1992
- 2 Scott Robert Ladd 著. C++ 技巧与应用. 阎龙等译. 北京:北京航空航天大学出版社,1992
- 3 田中秀和著. C++によるプログラム设计法. 东京:综合电子出版社,1992
- 4 河西朝雄著. C++. 东京:ナツメ社,1992
- 5 迪克,曾霞,李莹编译. C++实用工具箱编制技术与实例. 北京:北京希望电脑公司,1991
- 6 TOM SWAN 著. 汤姆·斯旺 C++ 编程秘诀. 宋建云等译. 北京:电子工业出版社,1994
- 7 施小龙,葛玉宝,邓明辉编. Borland C++ 深入编程. 北京:学苑出版社,1994
- 8 李长江,赵志辉,陈杰编. C++ 使用手册. 北京:电子工业出版社,1995

目 录

第一章 过程初步	1	3.11 函数模板	112
1.1 C++ 程序初识	1	第四章 类与对象	118
1.2 变量——程序员魔术师的道具	2	4.1 类的定义与对象的创建	118
1.3 表达式	4	4.2 构造体、位段和共用体	132
1.4 选择程序设计	7	4.3 对象的访问控制	136
1.5 穷举程序设计(重复结构应用之一)	14	4.4 对象与指针	139
1.6 递推与迭代程序设计(重复结构应用 之二)	17	4.5 链表类	143
1.7 程序测试用例设计	26	4.6 运算符重载	147
第二章 过程管理	32	4.7 动态字符串类	153
2.1 用函数组织过程	32	4.8 类模板	162
2.2 递归程序设计	36	第五章 继承与多态性	166
2.3 随机模拟程序设计(库函数应用之一)	38	5.1 基底与派生	166
2.4 简单图形程序设计(库函数应用之二)	49	5.2 集合类	182
2.5 变量的存储属性	56	5.3 数组类	192
2.6 编译预处理	59	5.4 多重继承与虚拟基类	199
第三章 数据类型	64	5.5 多态性与虚函数	204
3.1 基本类型	64	5.6 包容类	213
3.2 一维数组	65	第六章 I/O 流	219
3.3 排序	75	6.1 流的基本操作	219
3.4 多维数组	79	6.2 流的格式化	224
3.5 图	82	6.3 流的出错处理	230
3.6 指针基础	90	6.4 文件的输入和输出	232
3.7 队列与堆栈	98	附录 Turbo/Borland C++ 库函数	245
3.8 回溯法程序设计	104	附录 1 分类函数	245
3.9 复杂声明	110	附录 2 数学函数	246
3.10 枚举	111	附录 3 串和内存操作函数	250
		附录 4 输入输出函数	254
		附录 5 图形函数	261
		参考文献	272

第一章 过程初步

1.1 C++ 程序初识

程序的设计、运行都需要有一个相应的计算机语言支持环境。进行程序设计前首先要熟悉某种程序设计语言的符号系统与程序结构,并且,为在这种语言环境中实施程序设计,必须了解这种语言环境为程序的编辑、编译、连接、调试所提供的支持及其使用方法。

C(C++)程序的基本结构为

```
main( )
{
    ...    // 语句序列
}
```

程序中的每一个语句应以分号“;”结束。并且可以根据需要,在程序的前边或中间的任何位置插入以“#”打头的编译预处理命令,如 #include ...。为了便于阅读程序,还可以在C++ 程序中的任何位置插入以“//”打头的注释。

C++ 程序的基本输入、输出语句为

```
cin >> ...;
```

```
cout << ...;
```

“>>”、“<<”分别称为输入、输出运算符;“cin”、“cout”分别表示标准输入、输出设备——键盘、显示器等;“...”是输入、输出的内容,称为输入、输出表达式。

1.1.1 在你的系统上编辑并运行下面的 C++ 程序:

```
ex111.cpp
#include <iostream.h>
MAIN( ) { cout << "Hello,C++!" }
```

使其输出如下的一串字符:

```
Hello,C++!
```

提示

- (1) C++(C)语言对同一字母的大小写将区别为两个不同的字符;
- (2) C++(C)中每个语句应以分号结束。

1.1.2 在 Borland/Turbo C++ 系统上将源程序文件名后缀定义为“.C”与“.CPP”将会产生什么不同的后果? 为什么?

1.1.3 写一个 C++ 程序,使其输出如下串字符:

```
Programming is fun.
```

1.1.4 编写一个 C++ 程序,使其输出如下两行字符:

Programming is fun.

Hello,c++!

1.1.5 试运行下面的程序,并指出输出结果。

ex115.cpp

```
#include <iostream.h>
void main( )
{
    int x,y;
    cout << "Enter two numbers separated by space:" << endl;
    cin >> x >> y;
    cout << "x + y = " << x + y << endl;
}
```

1.1.6 若运行下面的程序,会输出什么?

ex116.cpp

```
#include <iostream.h>
void main( )
{
    int i = 0;
    while(i < 5) {
        cout << "Hello,c++! \n";
        i++;
    }
}
```

1.1.7 下面各小题例举的是 C++ 的合法注释吗?为什么?

- (1) /* This is an attepte /* to next */ a comment */
- (2) // This is an attepte to next a comment//
- (3) /* This is an attepte to next a comment */

1.2 变量——程序员魔术师的道具

变量标识了程序中可命名的内存空间。程序对数据的处理,就是程序在执行过程中通过对变量值的不断更新完成的。对程序员来说,变量是他进行魔术表演的道具。正确地使用变量,是程序员的最基本训练之一。

C++ 要求,在程序中使用一个名字之前,先要声明它的类型。使用一个变量名之前,必须对它进行声明。声明与定义不同,定义是为变量分配存储空间,而单纯的声明无此功能。不过,有一些声明也具有定义功能。

C++ 变量名是用户定义的标识符,其命名原则因系统而异。

1.2.1 下列单词中,哪些是合法的 C++ 变量名?

ABC 2abc π $\alpha\beta\gamma$ HuabeiEconomicUniversity a.b
 swap# -123 _ab a - b a_b main a__b ap(a,b)

提示 Borland/Turbo C++ 按 DOS 的规定命名变量名:

- ① 组成字符:大小写字母、数字、下划线;
- ② 原则上为任意长度,但 Borland/Turbo C++ 只将前 32 个字符看作有效;
- ③ 以英文字母或下划线开头;
- ④ 不可使用系统保留字和保留字符,如 C 保留以下划线为一些标识符的起始标志,C++ 也保留含有双下划线的标识符,所以用户应避免使用含有双下划线或以下划线作为起始的标识符。

1.2.2 指出下面的程序中每一个语句执行后各变量的值。

ex122.cpp

```
#include <iostream.h>
void main( )
{
    int x = 2, y = 3;
    x = x - y;
    y = x + y;
    x = y - x;
    cout << "x=" << x << ", y=" << y << endl;
}
```

1.2.3 指出下面程序中的错误。

ex123.cpp

```
#include <iostream.h>;
void main( )
{
    const: x = 3;
    y = 4;
    x = x + y;
    cout << x;
}
```

1.2.4 请指出下列声明中变量 a,b,c 的异同。

```
int a = 1;
const int b = 1;
const c = 1;
```

1.2.5 请指出下列声明中变量 a,b,c,d 的异同。

```
int a = 0;
const int b = 0;
volatile int c = 0;
```

```
const volatile d = 0;
```

提示 当一小段程序多次使用同一变量时,编译器将把其置成寄存器变量,以提高操作的效率。若将变量声明为 `volatile`,将丧失这一功能。

1.3 表 达 式

琳琅满目的运算符与机动灵活的表达式,是 C++ (C) 语言的重要特征之一,也是初学者感到困难的一点。表 1.1 为 C++ 的运算符一览表。

表 1.1 C++ 的运算符

优先级	运 算 符	结合性
1	() -> [] :: .(分量)	→
2	* (递引用) & (取地址) new delete ! ~ ++ -- - (单目减) sizeof	←
3	* / %	→
4	+ -	
5	<< >>	
6	< <= > >=	
7	= = ! =	
8	& (引用)	
9	^	
10		
11	&&	
12		
13	?:	←
14	= += -= *= /= %= <<= >>= &= ^= =	
15	,	→

使用运算符与表达式应注意以下几点:

- (1) 运算对象的类型,每一种运算符都与一定的数据类型相联系;
- (2) 优先级与结合性;
- (3) 避免表达式的副作用。

1.3.1 下列两个 C++ 表达式的值各是什么? 这两个 C++ 表达式的计算规则与数学中的计算规则有何异同? 为什么?

- (1) $1000 * 5 / 7$
- (2) $5 / 7 * 1000$

提示

(1) 数学中的运算规则按运算符的优先级、交换律、结合律进行。在 C++ 中,则要按照优先级、结合性以及机器的实现方式决定计算规则。

(2) 在 C++ 中每一种运算符都与特定的数据类型相联系。

1.3.2 在 1.3.1 的两个表达式中各修改一个数据,使它们的运算结果相同。

1.3.3 执行下面的语句

```
char c = 'w';
int i = 1, j = 2, k = -7;
float x = 700000000000, y = 0.001;
```

后,下面各表达式的值是什么?

- | | |
|------------------------|-------------------------|
| (1) 'a' + 1 < c | (2) -i - 5 * j >= k + 1 |
| (3) 3 < i < 5 | (4) x - 3.33 <= x + y |
| (5) x < x + y | (6) 'v' == c - 1 |
| (7) i + j + k = -2 * j | (8) k == j - 9 == i |
| (9) !(-i - j >= 0) | (10) x != x + y |

提示 浮点数不像整型数那么精确,任何涉及浮点运算的值都会有一个小的误差,误差的大小取决于机器字长。

1.3.4 执行下面的语句

```
char c = 'w';
int i = 0, j = 0, k = 0;
float x = 0.0, y = 2.3;
```

后,下面各表达式的值是什么?

- | | |
|--------------------------------|----------------------------------|
| (1) ! x * ! y | (2) !!! x |
| (3) x i && j - 3 | (4) i < j && x < y |
| (5) i < j x < y | (6) i == j && x <= y |
| (7) x != y && j + 1 == k + 4 | (8) 'A' <= c && c <= 'Z' |
| (9) -- C == 'v' ++ C == 'v' | (10) i == z j == 4 (k = 6) |
| (11) !(i < j && x < y) | (12) !(i == j x <= y) |

1.3.5 用 C++ 语言描述下面的命题:

- | | |
|----------------------|-----------------|
| (1) a 小于 b 或小于 c | (2) a 或 b 都大于 c |
| (3) a 和 b 中有一个小于 c | (4) a 是奇数 |
| (5) a 是非正整数 | (6) a 不能被 b 整除 |
| (7) a 与 b 是两个符号相反的实数 | |

1.3.6 A, B, C 是三个逻辑变量,写出 A, B, C 相等的条件。

解 求逻辑表达式值的基本方法为真值表法。表 1.2 为题中 A, B, C 三个逻辑变量的真值表。由真值表可以看出,在下边两种情形下, A 与 B 与 C 相等:

A, B, C 全为 0 (“假”);

A, B, C 全为 1 (“真”)。

用 C++ 逻辑表达式描述为

$$A \parallel B \parallel C == 0$$

或

$$A \&\& B \&\& C == 1$$

即

$$(A \parallel B \parallel C == 0) \parallel (A \&\& B \&\& C == 1) == 1$$

由于 A, B, C 为逻辑变量, 所以也可以写成

$$(! A \&\& ! B \&\& ! C) \parallel (A \&\& B \&\& C)$$

1.3.7 某宿舍中住着 A, B, C, D 四个人:

- (1) A 从来不唱歌;
- (2) 只有 A 在场时 B 和 D 才唱歌;
- (3) C 在任何情况下都不唱歌。

请根据上面的条件, 写出“宿舍里没人唱歌”的 C++ 逻辑表达式。

1.3.8 修改下面程序中的错误, 并指出输出结果。

ex138.cpp

```
void main( )
{
    int x = y = z = 1;
    y = y + z;
    x = x + y;
    cout << x < y ? y : x << endl;
    cout << x > y > x ? 1 : 0 << endl;
    cout << z >= y && y = x << endl;
    x = y ? cout << x : x ++;
}
```

1.3.9 对语句

int x = 3, y = 2, z = 1;

计算下列各式的值。

- | | |
|--------------------------|-------------------------|
| (1) $x \parallel y \& z$ | (2) $x \parallel y - z$ |
| (3) $x \wedge y \& -z$ | (4) $x \& y \&\& z$ |

1.3.10 对语句

int x = 1, y = -1;

计算下列各式的值。

- | | |
|-----------------------|--------------------------|
| (1) $! x \parallel x$ | (2) $\sim x \parallel x$ |
| (3) $x \wedge x$ | (4) $y << 2$ |

1.3.11 对于语句

int a = 2, b = 2, c = 2;

判断下列表达式是否正确, 并计算正确的表达式的值。

- | | |
|---------------------------------|--------------------------------|
| (1) $a = b + + + c + +$ | (2) $a = + + b + + + c$ |
| (3) $a = + + b + c + +$ | (4) $a = b - - + - - c$ |
| (5) $a = b - - - c$ | (6) $a = -b + + + c$ |
| (7) $a = b + + + + + c$ | (8) $a < < = 2$ |
| (9) $a = (b - = 3) + (c + = 5)$ | (10) $a + = b - = c / = a - 3$ |

提示 请向机器请教。

1.3.12 对于语句

```
int a = 5, b = 5, c = 5;
```

下面的表达式有何不妥之处,应如何修改?

- | | |
|---------------------------------|-----------------------------|
| (1) $c = + + a + a$ | (2) $c = b - - + b$ |
| (3) $c = (a = b + 2) + (b = 7)$ | (4) $c = (a + 3) / 0 + + a$ |

1.4 选择程序设计

选择赋予程序以“智能”。具有选择能力的程序,才能根据不同情况采取相应的对策。C (C++) 主要用 if-else 结构实现选择。其基本结构形式为

```
if (<表达式>)
    语句1
else
    语句2
```

当表达式的值为“非 0”(“真”)时,执行“语句₁”;当表达式的值为“0”(“假”)时,执行“语句₂”。这样,就形成程序的二路分支结构。

采用 if-else 的嵌套结构形式,也可以形成程序的多路分支结构:

```
if (<表达式1>)
    语句1
else if (<表达式2>)
    <语句2>
    ...
else
    <语句n>
```

这里,“表达式_i”是当“表达式_{i-1}”不满足时的进一步判断;满足“表达式_i”时,将执行“语句_i”。

多路分支结构的另一种形式是 switch 结构:

```

switch (〈整型表达式〉) {
    case 〈常量表达式1〉:
        〈语句1〉
    case 〈常量表达式2〉:
        〈语句2〉
        ...
    case 〈常量表达式n〉:
        〈语句n〉
    default:
        〈语句n+1〉
}

```

if-else 的嵌套结构与 switch 结构的重要区别在于测试条件不同。前者测试的是二值逻辑表达式,每一个 if 后面的逻辑表达式都只产生“真”(非 0)或“假”(0)二者之一。switch 结构中只有一个表达式,这个表达式只能是整型表达式。它的可取值多,每个 case 后都给出了一个预先估计的值,其后是针对该情形所对应的操作。当某一 case 值与 switch 表达式匹配时,便以此 case 作为该 switch 结构的入口。为了能在执行完某一个 case 的语句(可能是复合语句)后,便退出 switch 结构,应在该语句后加一条 break 语句。

1.4.1 分析并指出下面程序的功能。

ex141-1.cpp

```

#include <iostream.h>
void main( )
{
    int a,b,c,max;
    cout << "Enter 3 integers separated by spaces;" << endl;
    cin >> a >> b >> c;
    cout << "The max is:";
    max = a;
    if(c > b)
        if(c > a)
            max = c;
    else
        if(b > a)
            max = b;
    cout << max << endl;
}

```

解 这个试图从 3 个数中取最大数的程序是一个错误的程序。错在什么地方呢? 尽管程序设计者采用了缩进格式,但编译器并不承认设计者的思路,而是按从前向后,一一将每个 else 与其前最近的、尚未被配对的 if 匹配的原则来进行语法分析,把它当作下面的程序来看

待:

```

ex141_2.cpp
#include <iostream.h>
void main( )
{
    int a,b,c,max;
    cout << "Enter 3 integers separated by spaces:" << endl;
    cin >> a >> b >> c;
    cout << "The max is:";
    max = a;
    if(c > b)
        if(c > a)
            max = c;
        else
            if(b > a)
                max = b;
    cout << max << endl;
}

```

由于错误的缩进格式,程序具有很大的迷惑性,因此应特别注意以下两点:

- (1) 要按语法规定的匹配关系写缩进格式;
- (2) 尽量不使用 if 与 if...else 混合嵌套结构。非使用不可时,可以采用花括号强制编译器按人的意图去理解程序。如:

```

ex141_3.cpp
#include <iostream.h>
void main( )
{
    int a,b,c,max;
    cout << "Enter 3 integers separated by spaces:" << endl;
    cin >> a >> b >> c;
    cout << "The max is:";
    max = a;
    if( c > b )
    {
        if( c > a )
            max = c;
    }
    else
    {
        if( b > a )
            max = b;
    }
}

```

```
    |  
    cout << max << endl;  
    |
```

1.4.2 阅读下面的程序,指出程序的功能。

(1)

ex142_1.cpp

```
#include <iostream.h>  
void main( )  
{  
    const int magic = 17;  
    int guess;  
    cout << "guess the magic number:"  
    cin >> guess;  
    if(guess == magic)  
    {  
        cout << " * * Right * *" << endl;  
        cout << "The magic number is:" << magic;  
    }  
    else  
    {  
        cout << " * * wrong * *" << endl;  
        if(guess > magic) cout << "Too high";  
        else               cout << "Too low";  
    }  
}
```

(2)

ex142_2.cpp

```
#include <iostream.h>  
int math,chem,phy,ave;  
void main( )  
{  
    cout << "Enter the scores:" << endl;  
    cin >> math >> chem >> phy;  
    ave = (math + chem + phy) / 3.0 + 0.5;    // 0.5 的作用是什么?  
    if(ave >= 90) cout << "Excellent";  
    else  
        if(ave >= 70 && ave < 90)    cout << "Average";  
    else  
        if(ave >= 60 && ave < 70)    cout << "Pass";
```